# Three Permutations

# If your task is to exhaustively generate all possibilities, consider using recursion. Here are three small programs that permute letters or digits.

|  |
| --- |
| LLL |
| LLR |
| LRL |
| LRR |
| RLL |
| RLR |
| RRL |
| RRR |

## Left Right

Write a program that prompts for a positive integer and displays all permutations of the letters "L" and "R" of that length. If the length is 3 your program should display 🡪

## This problem is naturally recursive. Start with an empty string. Recursively generate all the possibilities. When the string is large enough, print. 🡨clue

## public static void leftRight(String s, int n)

## Odd Digits

|  |
| --- |
| 111 |
| 113 |
| 115 |
| 117 |
| 119 |
| 131 |
| 133 |
| 135 |
| **. . .** |

Write a program that prompts for a positive integer and displays all permutations of the odd digits of that length. Display these values in ascending order. For example, if the length is 3, your program should display 🡪

## Super Prime

A superprime number is a prime number with this additional quality: all of the numbers formed by repeatedly chopping off the last digit of the number are also prime. For example, if

2939 is prime AND

293 is prime AND

How many digits? 4  
2333  
2339  
2393  
2399  
2939  
3119  
3137  
3733  
3739  
3793  
3797  
5939  
7193  
7331  
7333  
7393

29 is prime AND

2 is prime, THEN

2939 is superprime.

Write a program that prompts for a positive integer and displays all superprime numbers of that length. Display these values in ascending order. Recall that by definition, 2 is a prime number, and 1 is NOT a prime number. At the right is a sample run.

Hint: It is easier to generate superprimes by building them up. Use recursion to generate all possibilities. Test each possibility with the isPrime method. Keep the ones that pass the test.

**Extensions – 1/2 pt each**

1. Count the number of superprimes generated. Use a static variable.

2. Make isPrime run as fast as it can.